06-20-00

PATENT

Docket No. 6169-159

PATENT APPLICATION TRANSMITTAL LETTER

Commissioner for Patents
BOX PATENT APPLICATION
Washington, DC 20231

Transmitted herewith for filing of the patent application of:

Inventors: Brett Gavagni, Sunrise, FL
Bruce D. Lucas, Yorktown Heights, NY

For: SECURED ENCRYPTED COMMUNICATIONS IN A VOICE BROWSER

Enclosed are:

- X Specification, including the Abstract
- 1 Sheet of drawings (3 sets)
- X Executed Declaration and Power of Attorney
- X Executed Assignment and Recordation Cover Sheet
- X Other: 2 postcards

CLAIMS AS FILED

FOR	NO. FILED	NO. EXTRA
Basic Fee		
Total Claims	-40-	20
Indep Claims	-4-	1
multiple dependent claim present No		

If the difference in Col. 1 is less than zero, enter "0" in Col. 2

Small Entity		Other than a Small Entity	
RATE	FEE	RATE	FEE
	$ 345		$ 690
x $ 9 =	$	x $ 18 =	$ 360
x $ 39 =	$	x $ 78 =	$ 78
x $130 =	$	x $260 =	$
TOTAL	$	TOTAL	$1,128

Assignment Recordal Fee $ 40

___ Please charge my Deposit Account No. 17-0055 in the amount of $ _____.
X A check in the amount of $1,168.00 is enclosed.
X The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No. 17-0055. A duplicate of this sheet is enclosed.
- X Any additional filing fees required under 37 C.F.R. 1.16.
- X Any patent application processing fees under 37 C.F.R. 1.17.
- __ No fee enclosed. No fee is authorized.

6.19.00
Date

Steven M. Greenberg
Registration No. 44,725

"EXPRESS MAIL" LABEL NO.: EK575132635US

QBWPB\161424.1

SECURED ENCRYPTED COMMUNICATIONS IN A VOICE BROWSER

Inventor(s): Brett Gavagni
Bruce D. Lucas

International Business Machines Corporation

IBM Docket No. BOC9-2000-0014
IBM Disclosure No. BOC8-2000-0019

Express Mail Label No. EK575132635US

BACKGROUND OF THE INVENTION

Technical Field

This invention relates to the field of computer audio user interfaces and more particularly to a system and method for performing secured communications between a voice browser and a server.

Description of the Related Art

Privacy in data communications has become a significant issue with the exponential increase in e-commerce transactions on the Internet. Typically a client computer and server computer require that exchanged information remain private to both parties. For instance, in an online banking transaction, the client requires that the sharing of the client's account number and password include only the intended bank and no other party. Presently, privacy in data transactions can be secured only in selected applications protocols through the use of security technologies which can incorporate either asymmetric, symmetric or a combination of asymmetric and symmetric encryption algorithms. The Secured Sockets Layer ("SSL") protocol represents one such security technology which incorporates both asymmetric and symmetric encryption algorithms.

SSL is a transport-layer protocol that can be established between a client and a server. SSL is typically integrated directly with selected underlying application protocols. For example, the Hypertext Transfer Protocol ("HTTP") has been successfully integrated with SSL. Specifically, HTTP packets are encapsulated in SSL packets and are transported over TCP/IP. HTTP integrated with SSL is commonly referred to as "HTTPS" and can be used to securely view and exchange Web-based content encoded in hypertext markup language ("HTML"). Other protocols integrated with SSL include Telnet, the File Transfer Protocol ("FTP"), the Lightweight Directory Access Protocol ("LDAP"), the Internet Message Access Protocol ("IMAP"), and the Network News Transfer Protocol ("NNTP").

SSL is intended to provide a secure pipe between a client and a server. SSL is

session-oriented and can maintain state, despite the execution of SSL over such protocols as HTTP which, in of itself, is stateless. Finally, SSL provides privacy through encryption, both asymmetric and symmetric, authentication based upon certificates, a vehicle for authorization through SSL's support for certificates, integrity by incorporating hash functions, and digital signing as part of the transport protocol.

Briefly, in an SSL compliant visual Web browser executing the "HTTPS" protocol, an SSL session can be established when a client selects a uniform resource locator ("URL") referencing a server compliant with the HTTPS protocol. The server can respond by delivering to the client, an X.509 certificate containing a distinguished name referencing a Certificate Authority ("CA") and a public key. The client can examine the server certificate by referencing the issuing CA and can verify the integrity of the server certificate if the issuing CA is configured in the visual Web browser as trustworthy. Subsequently, the server can perform optional client authentication by requesting a certificate from the client. The server, too, can examine the client certificate by referencing the issuing CA and can verify the integrity of the client certificate if both the client and the issuing CA are configured in the server as trustworthy. If the server successfully authenticates the client certificate, the SSL session can continue. Otherwise , the session can be terminated.

Subsequently, the client can "challenge" the server using asymmetrical encryption technology in order to verify that the server indeed possesses the private key associated with the public key contained in the server certificate. In challenging the server, the client can generate a random string of data and can encrypt the random string of data using the server's public key contained in the server certificate. The client can transmit the encrypted data to the server and can request that the server deliver the data to the client. In order to deliver the data to the client, however, the server first must decrypt the data using the server's private key which corresponds to the server's public key contained in the server certificate. Optionally, the server, too, can challenge the client using a similar exchange of encrypted data.

Once the client and server have been mutually authenticated, the client and the server can agree upon a shared secret for use in future symmetrical encryption and decryption operations. Typically, the client can select the secret and encrypt the selected secret using the server's public key. The client can transmit the asymmetrically encrypted secret to the server so that only the client and the server share the common secret. When both the client and the server have agreed upon the shared secret, symmetrical data transfer can begin between the client and the server using the shared secret as the key to the symmetrical encryption and corresponding decryption operations. Notably, a more thorough treatment of the SSL protocol has been published by Netscape Communications Corporation of Mountain View, California in Freier, Karlton, Kocher, *The SSL Protocol Version 3.0* (Netscape Communications Corp. March 1996), incorporated herein by reference. Additionally, an SSL 3.0 compatible standard has been approved by the Internet Engineering Task Force ("IETF") and has been published by the IETF as Dierks & Allen, *RFC2246 - The TLS Protocol Version 1.0* (IETF January 1999), incorporated herein by reference.

Unlike visual Web browsers executing the HTTPS protocol, SSL has not been integrated with Voice Browsers. Generally, a Voice Browser, unlike a visual Web browser, does not permit a user to interact with Web-based content visually. Rather, a Voice Browser, which can operate in conjunction with a Speech Recognition Engine and Speech Synthesis Engine, can permit the user to interact with Web-based content audibly. That is, the user can provide voice commands to navigate from Web-based document to document. Likewise, Web-based content can be presented to the user audibly, typically in the form of speech synthesized text. Thus, Voice Browsers can provide voice access and interactive voice response to Web-based content and applications, for instance by telephone, personal digital assistant, or desktop computer.

Significantly, Voice Browsers can be configured to interact with Web-based content encoded in VoiceXML. VoiceXML is a markup language for distributed voice applications based on extended markup language ("XML"), much as HTML is a markup

language for distributed visual applications. VoiceXML is designed for creating audio dialogs that feature synthesized speech, digitized audio, recognition of spoken and Dual Tone Multifrequency ("DTMF") key input, recording of spoken input, telephony, and mixed-initiative conversations. Version 1.0 of the VoiceXML specification has been published by the VoiceXML Forum in the document Linda Boyer, Peter Danielsen, Jim Ferrans, Gerald Karam, David Ladd, Bruce Lucas and Kenneth Rehor, *Voice eXtensible Markup Language (VoiceXML™) version 1.0*, (W3C May 2000), incorporated herein by reference. Additionally, Version 1.0 of the VoiceXML specification has been submitted to the World Wide Web Consortium by the VoiceXML Forum as a proposed industry standard.

Version 1.0 of the VoiceXML specification provides a high-level programming interface to speech and telephony resources for application developers, service providers and equipment manufacturers. As noted in W3C submission, standardization of VoiceXML will simplify creation and delivery of Web-based, personalized interactive voice-response services; enable phone and voice access to integrated call center databases, information and services on Web sites, and company intranets; and help enable new voice-capable devices and appliances. Still, the VoiceXML specification lacks a mechanism for secure communications through encrypted network transmissions via the SSL protocol over the TCP/IP protocol. Accordingly, what is needed is a Voice Browser incorporating SSL support for performing secure communications in a data communications network.

SUMMARY OF THE INVENTION

The present invention is a Voice Browser for processing VoiceXML encoded Web content through a secure connection established using symmetric and asymmetric encryption techniques. In the preferred embodiment, the symmetric and asymmetric encryption techniques are included in a Java implementation of the SSL 3.0 protocol for providing secure communications through encrypted network transmissions between a VoiceXML-compliant Voice Browser Server and a network device. Specifically, the method of the present invention can authenticate the network device and negotiate a shared secret between the client and the server using asymmetrical encryption techniques. Subsequently, the method of the present invention can facilitate secure communications between the client and the server of data in a VoiceXML document using symmetrical encryption techniques.

The method of the invention can include the steps of transmitting a request to the network device to establish a secured communication session between the Voice Browser and the network device and authenticating the network device. Subsequent to the authentication, a shared secret can be negotiated between the network device and the Voice Browser. Once a shared secret has been negotiated, VoiceXML-based Web content can be encrypted using the shared secret as an encryption key. Additionally, the encrypted VoiceXML-based Web content can be exchanged between the network device and the Voice Browser. Finally, the VoiceXML-based Web content can be decrypted using the shared secret as a decryption key. Significantly, the Voice Browser can be a VoiceXML Browser Server.

The step of authenticating the network device can include transmitting a digital certificate from the network device to the Voice Browser and validating the certificate authority. The digital certificate can have a public key and a reference to a certificate authority. Specifically, the digital certificate can be an X.509-compliant digital certificate. Optionally, the method can further include the step of authenticating the Voice Browser. The step of authenticating the Voice Browser can include transmitting a

digital certificate from the Voice Browser to the network device and validating the certificate authority. As before, the digital certificate can have a public key and a reference to a certificate authority. Specifically, the digital certificate can be an X.509-compliant digital certificate.

The step of authenticating the network device can further include the step of challenging the network device. Likewise, the step of authenticating the Voice Browser can further include the step of challenging the Voice Browser. The step of challenging the network device can include encrypting a message using the public key contained in the digital certificate; transmitting the encrypted message from the Voice Browser to the network device; decrypting the encrypted message using a private key corresponding to the public key; and, transmitting the decrypted message to the Voice Browser. Similarly, the step of challenging the Voice Browser can include encrypting a message using the public key contained in the digital certificate; transmitting the encrypted message from the network device to the Voice Browser; decrypting the encrypted message using a private key corresponding to the public key; and, transmitting the decrypted message to the network device.

In the preferred embodiment, the negotiating step can include the steps of: generating a key for use in a symmetric cryptographic algorithm; encrypting the generated key with the public key; transmitting the encrypted key to the network device; and, decrypting the key in the network device with a private key corresponding to the public key. Alternatively, the negotiating step can include the steps of: generating a key for use in a symmetric cryptographic algorithm; encrypting the generated key with the public key; transmitting the encrypted key to the Voice Browser; and, decrypting the key in the Voice Browser with a private key corresponding to the public key.

In the preferred embodiment, the method of the present invention can further include the steps of: exchanging a list of supported symmetrical cryptographic algorithms for the network device and the Voice Browser; selecting a symmetrical cryptographic algorithm from the list; and, performing the encrypting and decrypting

steps using the selected symmetrical cryptographic algorithm.

A method for performing secured communications in a Voice Browser can include the steps of: transmitting a request from the Voice Browser to a network device for a secure communications session between the Voice Browser and the network device; receiving from the network device a digital certificate containing a public key and a reference to a certificate authority; and, authenticating the network device based on the digital certificate. Preferably, the digital certificate can be an X.509-compliant digital certificate.

Subsequent to the authentication, the method can include the steps of negotiating a shared secret with the network device; encrypting data using the shared secret as an encryption key and transmitting the encrypted data to the network device; and, receiving encrypted Web content from the network device and decrypting the Web content using the shared secret as a decryption key. Significantly, the Web content can be a VoiceXML document and the Voice Browser can be a VoiceXML Browser Server.

In the preferred embodiment, the transmitting step can further include the steps of: transmitting to the network device a list of supported encryption algorithms for use in the encryption and decryption steps. Notably, the network device can select an encryption algorithm from among the list. Subsequently, the data can be encrypted using the selected encryption algorithm and the Web content can be decrypted using the encryption algorithm.

BRIEF DESCRIPTION OF THE DRAWINGS

There are presently shown in the drawings embodiments which are presently preferred, it being understood, however, that the invention is not limited to the precise arrangements and instrumentalities shown.

Fig. 1 is an illustration of the establishment of a secured communications session between a Voice Browser and a network device.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is a Voice Browser enabled to perform secured communications with a network device. In particular, the Voice Browser can request a secured connection with a network device. Subsequently, the Voice Browser can receive a response from the network device in which the network device can acknowledge the request of the Voice Browser. Upon receiving the acknowledgment from the network device, the Voice Browser can authenticate the network device in order to ensure the identity of the network device. If the Voice Browser determines the identity of the network device to be authentic, the Voice Browser and the network device can select a shared secret to be used as an encryption key during an ensuing secured communications session. Finally, the Voice Browser and the network device can perform secured communications using the shared secret as an encryption key.

Advantageously, the secured communications functionality provided to the Voice Browser can be the result of the combination of a secured communications interface in the Voice Browser and a platform-independent, standards-based implementation of the Secured Sockets Layer ("SSL") secured communications protocol. In the preferred embodiment, the standards-based, platform-independent implementation of the SSL protocol is the SSLite for Java™ SSL implementation library manufactured by IBM Corporation of Armonk, New York. A class hierarchy for SSLite is attached hereto in Appendix A. Additionally, Javadocs documentation for each class listed in Appendix A are provided in Appendixes B-E. Specifically, Appendix B describes the class HttpsURLConnection, Appendix C describes the class HttpsClient, Appendix D describes the class HttpsURLStreamHandlerFactory and Appendix E describes the class HttpsURLStreamHandler. Still, the invention is not limited in regard to the particular secured communications library to be combined with the secured communications interface. In particular, the present invention can also incorporate the Transport Layer Security ("TLS") protocol defined by the Transport Layer Security Working Group of the Internet Engineering Task Force, the Kerberos protocol

developed by the Massachusetts Institute of Technology, or other suitable secured communications protocols.

Figure 1 illustrates a simplified approach to SSL secured communications between a Voice Browser and a network device. As shown in Figure 1, First, for an SSL connection to become established between the Voice Browser and the network device, an SSL handshake is performed. Specifically, the Voice Browser can transmit to the network device a "client hello" message. The client hello message can include a request for a connection with the network device in addition to the capabilities of the client, for example the preferred secured communications protocol, the cipher suites available to perform encryption and supported data compression methods. The network device can acknowledge the client hello message with a "server hello" message which can include a cipher suite selected from the cipher suites listed in the client hello message, and a compression method selected from the list of supported encryption methods. Notably, if the network device is unable to support any of the encryption algorithms contained in the cipher suite provided by the Voice Browser, the network device can notify the Voice Browser that the handshake attempt has failed. Subsequently, the connection between the Voice Browser and the network device can be closed.

Still, if handshake attempt is successful, the network device can transmit to the Voice Browser a digital certificate which can contain the network device's public key in addition to a reference to a certificate authority which acts as a trusted repository for digital certificates. The Voice Browser can authenticate the digital certificate by verifying that the certificate authority is a trusted repository of digital certificates. If the Voice Browser can successfully authenticate the certificate authority, a secure connection can be established. Notably, the network device can optionally authenticate the Voice Browser in the same the Voice Browser authenticates the network device. Notwithstanding, mutual authentication is not required in the present invention and the scope of the present invention is not to be limited in this regard.

Once the authentication process has been completed, the Voice Browser can transmit a "ClientKeyExchange" message to the network device. Specifically, the ClientKeyExchange message is a shared secret which has been encrypted using the public key of the network device, received in the digital certificate of the network device. The shared secret can be a randomly generated key for use in a symmetrical encryption algorithm. Despite the generation of the random key, however, the network device still preferably verifies that an identical key is not already in use with another client, be it another Voice Browser or other client application. If the network device determines that the key is already in use, the network device can notify the Voice Browser that another key must be generated. Notably, the invention is not limited in regard to the mechanism for generating a key. Rather, the key can be predetermined and stored in a database, generated according to a pre-defined algorithm, or other suitable key generation or key selection method.

When the Voice Browser and the network device have agreed upon a shared secret, specifically a common symmetric key for encrypting subsequently communications, both the Voice Browser and the network device can exchange a "ChangeCipherSpec" message confirming that both are ready to begin secured communications. Subsequently, the Voice Browser and the network device can begin secure communications using a symmetrical encryption algorithm with the shared secret as the encryption key.

With regard to the particular implementation of the present invention in which the SSL secured communications protocol is combined with secured communications interface of the Voice Browser, the SSL secured communications library can contain an "HttpsURLConnection" object which can provide methods for performing secured communications with HTTP servers. A complete description of the HttpsURLConnection class is included in the Javadoc "Class HttpsURLConnection" attached hereto as Appendix B. As is apparent from the class hierarchy of Appendix A, the HttpsURLConnection class is derived from the Java extension HttpURLConnection.

Accordingly, the class HttpsURLConnection is a platform-independent, standards-based implementation of the SSL protocol.

The following is a source code listing for a preferred interface between the Voice Browser and the secured communications protocol, specifically SSL. As is apparent from the source code, the following steps are minimally performed in order to establish a secured connection to a network device. First, a URL object is defined and instantiated for a fully-qualified URL. Concurrently, a stream handler is established for handling data streams received from the fully-qualified URL. Second, an unsecured connection is established with a network device addressed by the URL in which the symmetrical encryption algorithm can be specified in addition to the compression method. Also, the authentication process can be performed and a shared secret negotiated. Third, a secure connection can be established using the shared secret as a key to the selected encryption method.

```
import java.io.*;
import java.util.*;
import java.net.*;
import com.ibm.speech.net.www.protocol.https.*;

//begin class VoiceXMLBrowserServer
public class VoiceXMLBrowserServer
{
   public static void main(String args[])
   {
      try
      {
         String fullQualURL = new String ("https://www.ibm.com/software/speech/vxmlpage.vxml");

         URL url;

         URL.setURLStreamHandlerFactory(new HttpsURLStreamHandlerFactory());

         //Create file for inputstream dump
         FileOutputStream fout = new FileOutputStream("fetched.vxml");

         //Create URL object
         url = new URL(fullQualURL);

         //Setup Connection
         HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
```

```
        //SSL Implementation Specific API Extensions.
        //Optional usage, defaults included with implementation distribution
        //(Documented usage in Javadoc API definition)
        conn.setKeyRingDatabase("ralvs6");      //Set keyring database to use.
                                                //Default is specified as provided with implementation.
        conn.setTimeout(3);                     //Set connection timeout in seconds.
        conn.setAsyncConnections = true;        //Set SSL messages to be processed asynchronously
                                                //by a dedicated thread.
        conn.setEnabledCompressionMethods("IBM_ZIP_SPEED");       //Set compression method.
        conn.setEnabledCipherSuites("  SSL_RSA_WITH_RC4_128_MD5
                                       SSL_RSA_WITH_RC4_128_SHA");

        //Set enabled cipher suites (Encryption Algorithms)

        HttpURLConnection API (Standard Java platform networking API)
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
        conn.setRequestProperty("accept", "text/vxml");

        //Initiate secure connection
        conn.connect();

        //Get inputstream and do something with it
        if (conn.getInputStream()!=null)
        {
           BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
           String line;
           while ((line = in.readLine()) != null)
           {
              line = line + "\n";
              fout.write(line.getBytes());
           }
        }
        //Close connection
        conn.disconnect();
     }
     catch (Exception e)
     {
        System.out.println("Error: " + e.getMessage());
        e.printStackTrace();
     }
  }
}
//end class VoiceXMLBrowserServer
```

The preferred design and implementation of the present invention can be performed entirely in the Java programming language so as to avoid platform

dependencies. As such, the preferred design and implementation of the present invention is an "Optional Package". Optional packages , formerly known as "standard" extensions or "extensions" are packages of Java classes and associated native code that application developers can use to extend the functionality of the core platform. The extension mechanism allows a Java virtual machine (VM) to use the optional-package classes in much the same way as the VM uses bootstrap classes. Like bootstrap classes, classes in optional packages do not have to be placed on the class path. Also, the extension mechanism provides a method for needed optional packages to be retrieved from specified URLs when they are not already installed in the Java 2 Runtime Environment or Java 2 SDK.

Optional packages are embodied in JAR files, in which every JAR file is a potential optional package. A JAR file can be made to play the role of an optional package in two ways: First, by being placed in a special location in the Java 2 Runtime Environment or Java 2 SDK directory structure - in which case it is an "installed" optional package; and second, by being referenced in a specified way from the manifest of the JAR file of an applet or application - in which case it is a "download" optional package. When the VM is searching for a class of a particular name, it will first look among the bootstrap classes. If it fails to find the desired class there, it will next look for the class among any installed optional packages. If it doesn't find the class among either the bootstrap classes or the installed optional packages, the VM will search among any download optional packages referenced by the application or applet. The VM only searches the class path if it fails to find a class among the bootstrap classes or optional package classes.

The preferred embodiment of the present invention provides an abstraction of the underlying complicated key exchange, handshake and encrypted data transmission associated with secure data communications. In consequence, a Voice Browser in accordance with the inventive arrangements can access the abstracted method of the present invention through a reference to a library including an implementation of the

performing secured data communications. Hence, the present invention addresses the problem of secured communications in a Voice Browser by providing a Voice Browser incorporating SSL support for performing secure communications in a data communications network.

The method of the invention can be realized in hardware, software, or a combination of hardware and software. Machine readable storage according to the present invention can be realized in a centralized fashion in one computer system, or in a distributed fashion where different elements are spread across several interconnected computer systems. Any kind of computer system or other apparatus adapted for carrying out the methods described herein is acceptable. A typical combination of hardware and software could be a general purpose computer system with a computer program that, when being loaded and executed, controls the computer system such that it carries out the methods described herein. The present invention can also be embedded in a computer program product which comprises all the features enabling the implementation of the methods described herein, and which when loaded in a computer system is able to carry out these methods. A computer program in the present context can mean any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following: (a) conversion to another language, code or notation; and (b) reproduction in a different material form.

CLAIMS

1. A method for performing secured communications between a Voice Browser and a network device, said Voice Browser and network device exchanging VoiceXML-based Web content comprising the steps of:

transmitting a request to the network device to establish a secured communication session between the Voice Browser and the network device;

authenticating the network device;

subsequent to said authentication, negotiating a shared secret between the network device and the Voice Browser;

encrypting the VoiceXML-based Web content using said shared secret as an encryption key;

exchanging the encrypted VoiceXML-based Web content between the network device and the Voice Browser; and,

decrypting the VoiceXML-based Web content using said shared secret as a decryption key.

2. The method of claim 1, wherein said step of authenticating the network device comprises the steps of:

transmitting a digital certificate from the network device to the Voice Browser, said digital certificate having a public key and a reference to a certificate authority; and,

validating said certificate authority.

3. The method of claim 2, wherein said digital certificate is an X.509-compliant digital certificate.

4. The method of claim 1, further comprising the step of authenticating the Voice Browser.

5. The method of claim 4, wherein said step of authenticating the Voice Browser comprises the steps of:

transmitting a digital certificate from the Voice Browser to the network device, said digital certificate having a public key and a reference to a certificate authority; and,

validating said certificate authority.

6. The method of claim 5, wherein said digital certificate is an X.509-compliant digital certificate.

7. The method of claim 2, wherein said step of authenticating the network device further comprises the step of challenging the network device.

8. The method of claim 5, wherein said step of authenticating the Voice Browser further comprises the step of challenging the Voice Browser.

9. The method of claim 7, wherein said step of challenging the network device comprises the steps of:

encrypting a message using said public key contained in said digital certificate;

transmitting said encrypted message from the Voice Browser to the network device;

decrypting said encrypted message using a private key corresponding to said public key; and,

transmitting the decrypted message to the Voice Browser.

10. The method of claim 8, wherein said step of challenging the Voice Browser comprises the steps of:

encrypting a message using said public key contained in said digital certificate;

transmitting said encrypted message from the network device to the Voice

Browser;

decrypting said encrypted message using a private key corresponding to said public key; and,

transmitting the decrypted message to the network device.

11. The method of claim 1, wherein said negotiating step comprises the steps of:

generating a key for use in a symmetric cryptographic algorithm;

encrypting said generated key with said public key;

transmitting said encrypted key to the network device; and,

decrypting said key in the network device with a private key corresponding to said public key.

12. The method of claim 1, wherein said negotiating step comprises the steps of:

generating a key for use in a symmetric cryptographic algorithm;

encrypting said generated key with said public key;

transmitting said encrypted key to the Voice Browser; and,

decrypting said key in the Voice Browser with a private key corresponding to said public key.

13. The method of claim 1, further comprising the steps of:

exchanging a list of supported symmetrical cryptographic algorithms for the network device and the Voice Browser;

selecting a symmetrical cryptographic algorithm from said list; and,

performing said encrypting and decrypting steps using said selected symmetrical cryptographic algorithm.

14. The method of claim 1, wherein said Voice Browser is a VoiceXML Browser Server.

15. A method for performing secured communications in a Voice Browser comprising the steps of:

transmitting a request from the Voice Browser to a network device for a secure communications session between the Voice Browser and the network device;

receiving from the network device a digital certificate containing a public key and a reference to a certificate authority.

authenticating the network device based on the digital certificate;

subsequent to said authentication, negotiating a shared secret with the network device;

encrypting data using said shared secret as an encryption key and transmitting said encrypted data to the network device; and,

receiving encrypted Web content from the network device and decrypting the Web content using said shared secret as a decryption key.

16. The method of claim 15, wherein said transmitting step further comprises the step of:

transmitting to said network device a list of supported encryption algorithms for use in said encryption and decryption steps,

said network device selecting an encryption algorithm from among said list.

17. The method of claim 16, wherein said data is encrypted using said selected encryption algorithm and said Web content is decrypted using said encryption algorithm.

18. The method of claim 15, wherein said digital certificate is an X.509-compliant digital certificate.

19. The method of claim 15, wherein said Web content is a VoiceXML document.

20. The method of claim 19, wherein said Voice Browser is a VoiceXML Browser Server.

21. A machine readable storage, having stored thereon a computer program for performing secured communications between a Voice Browser and a network device, said Voice Browser and network device exchanging VoiceXML-based Web content, said computer program having a plurality of code sections executable by a machine for causing the machine to perform the steps of:

transmitting a request to the network device to establish a secured communication session between the Voice Browser and the network device;

authenticating the network device;

subsequent to said authentication, negotiating a shared secret between the network device and the Voice Browser;

encrypting the VoiceXML-based Web content using said shared secret as an encryption key;

exchanging the encrypted VoiceXML-based Web content between the network device and the Voice Browser; and,

decrypting the VoiceXML-based Web content using said shared secret as a decryption key.

22. The machine readable storage of claim 21, wherein said step of authenticating the network device comprises the steps of:

transmitting a digital certificate from the network device to the Voice Browser, said digital certificate having a public key and a reference to a certificate authority; and,

validating said certificate authority.

23. The machine readable storage of claim 22, wherein said digital certificate is an X.509-compliant digital certificate.

24. The machine readable storage of claim 21, for further causing the machine to perform the step of authenticating the Voice Browser.

25. The machine readable storage of claim 24, wherein said step of authenticating the Voice Browser comprises the steps of:

transmitting a digital certificate from the Voice Browser to the network device, said digital certificate having a public key and a reference to a certificate authority; and,

validating said certificate authority.

26. The machine readable storage of claim 25, wherein said digital certificate is an X.509-compliant digital certificate.

27. The machine readable storage of claim 22, wherein said step of authenticating the network device further comprises the step of challenging the network device.

28. The machine readable storage of claim 25, wherein said step of authenticating the Voice Browser further comprises the step of challenging the Voice Browser.

29. The machine readable storage of claim 27, wherein said step of challenging the network device comprises the steps of:

encrypting a message using said public key contained in said digital certificate;

transmitting said encrypted message from the Voice Browser to the network device;

decrypting said encrypted message using a private key corresponding to said public key; and,

transmitting the decrypted message to the Voice Browser.

30. The machine readable storage of claim 28, wherein said step of challenging the

Voice Browser comprises the steps of:

encrypting a message using said public key contained in said digital certificate;

transmitting said encrypted message from the network device to the Voice Browser;

decrypting said encrypted message using a private key corresponding to said public key; and,

transmitting the decrypted message to the network device.

31. The machine readable storage of claim 21, wherein said negotiating step comprises the steps of:

generating a key for use in a symmetric cryptographic algorithm;

encrypting said generated key with said public key;

transmitting said encrypted key to the network device; and,

decrypting said key in the network device with a private key corresponding to said public key.

32. The machine readable storage of claim 21, wherein said negotiating step comprises the steps of:

generating a key for use in a symmetric cryptographic algorithm;

encrypting said generated key with said public key;

transmitting said encrypted key to the Voice Browser; and,

decrypting said key in the Voice Browser with a private key corresponding to said public key.

33. The machine readable storage of claim 21, for further causing the machine to perform the steps of:

exchanging a list of supported symmetrical cryptographic algorithms for the network device and the Voice Browser;

selecting a symmetrical cryptographic algorithm from said list; and,

performing said encrypting and decrypting steps using said selected symmetrical cryptographic algorithm.

34. The machine readable storage of claim 21, wherein said Voice Browser is a VoiceXML Browser Server.

35. A machine readable storage, having stored thereon a computer program for performing secured communications in a Voice Browser, said computer program having a plurality of code sections executable by a machine for causing the machine to perform the steps of:

transmitting a request from the Voice Browser to a network device for a secure communications session between the Voice Browser and the network device;

receiving from the network device a digital certificate containing a public key and a reference to a certificate authority.

authenticating the network device based on the digital certificate;

subsequent to said authentication, negotiating a shared secret with the network device;

encrypting data using said shared secret as an encryption key and transmitting said encrypted data to the network device; and,

receiving encrypted Web content from the network device and decrypting the Web content using said shared secret as a decryption key.

36. The machine readable storage of claim 35, wherein said transmitting step further comprises the step of:

transmitting to said network device a list of supported encryption algorithms for use in said encryption and decryption steps,

said network device selecting an encryption algorithm from among said list.

37. The machine readable storage of claim 36, wherein said data is encrypted using said selected encryption algorithm and said Web content is decrypted using said encryption algorithm.

38. The machine readable storage of claim 35, wherein said digital certificate is an X.509-compliant digital certificate.

39. The machine readable storage of claim 35, wherein said Web content is a VoiceXML document.

40. The machine readable storage of claim 39, wherein said Voice Browser is a VoiceXML Browser Server.

ABSTRACT

A method for performing secured communications in a Voice Browser can include the steps of: transmitting a request from the Voice Browser to a network device for a secure communications session between the Voice Browser and the network device; receiving from the network device a digital certificate containing a public key and a reference to a certificate authority; and, authenticating the network device based on the digital certificate. Preferably, the digital certificate can be an X.509-compliant digital certificate. Subsequent to the authentication, the method can include the steps of negotiating a shared secret with the network device; encrypting data using the shared secret as an encryption key and transmitting the encrypted data to the network device; and, receiving encrypted Web content from the network device and decrypting the Web content using the shared secret as a decryption key. Significantly, the Web content can be a VoiceXML document and the Voice Browser can be a VoiceXML Browser Server.

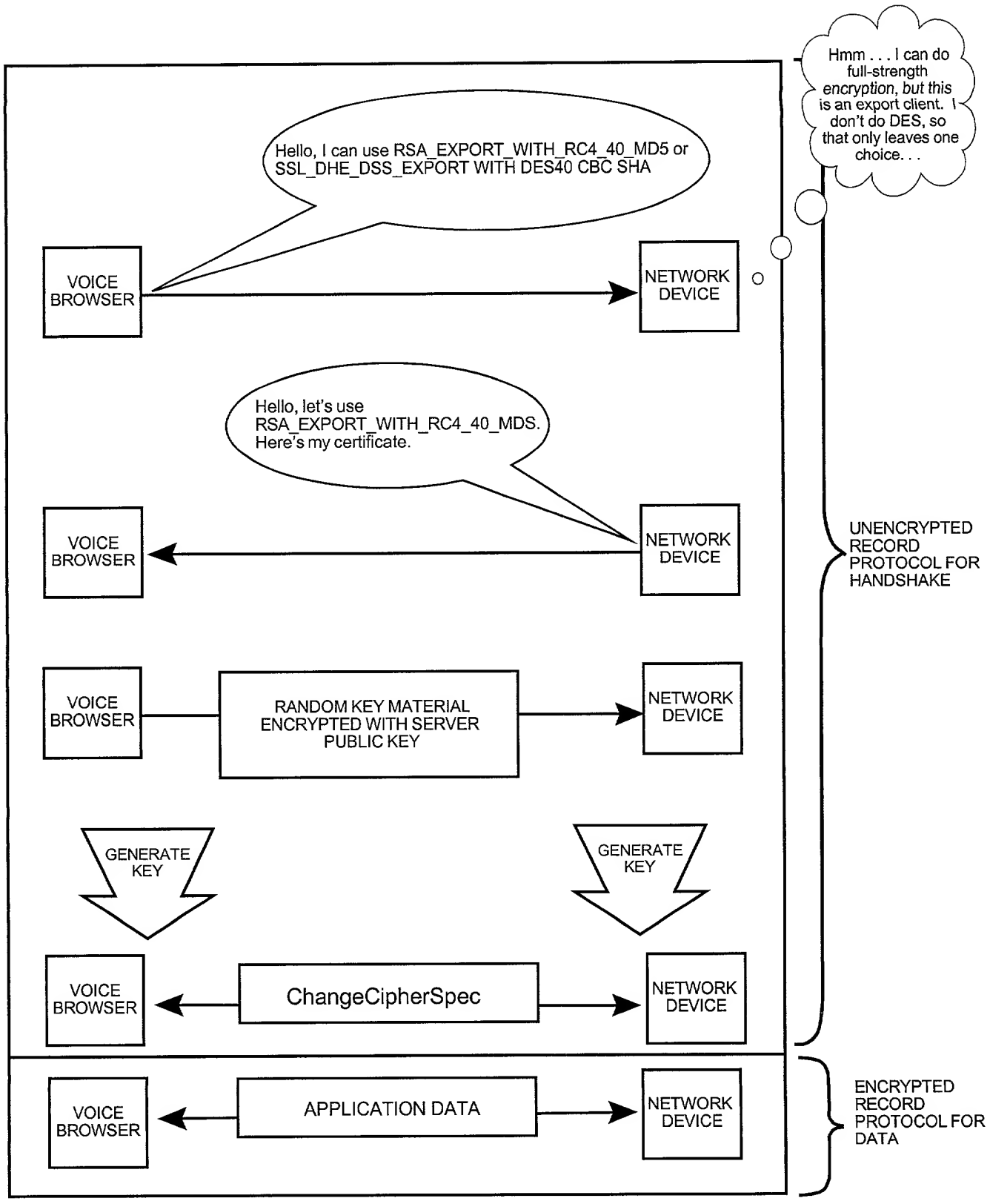

FIG. 1

DOCKET NUMBER: 6169-159

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am are the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

SECURED ENCRYPTED COMMUNICATIONS IN A VOICE BROWSER

the specification of which (check one)

_X_ is attached hereto.

____ was filed on _____
under Attorney's Docket Number _____
as Application Serial No. _____
and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations Section 1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)			Priority Claimed
_____ (Number)	_____ (Country)	_____ (Filing Date)	____ Yes ___No
_____ (Number)	_____ (Country)	_____ (Filing Date)	____ Yes ___No

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, we acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

_____ (Appln. Serial No.)	_____ (Filing Date)	_____ (Status)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Express Mail Label No. EK575132635US

DOCKET NUMBER 6169-159

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

J. Rodman Steele, Jr.	Reg. No. 25,931
Gregory A. Nelson	Reg. No. 30,577
Joseph W. Bain	Reg. No. 34,290
Robert J. Sacco	Reg. No. 35,667
Stanley Kim	Reg. No. 42,730
Mark D. Passler	Reg. No. 40,764
Steven Greenberg	Reg. No. 44,725

Send correspondence to Gregory A. Nelson, Quarles & Brady LLP, 222 Lakeview Avenue, Fourth Floor, P.O. Box 3188, West Palm Beach, Florida 33402-3188 and direct all telephone calls to Gregory A. Nelson at (561) 653-5000.

FULL NAME OF INVENTOR: Brett Gavagni

INVENTOR'S SIGNATURE: _____ DATE: 6/16/2000

RESIDENCE: 2932 NW 99th Terrace
Sunrise, FL 33322

CITIZENSHIP: U.S.A.

POST OFFICE ADDRESS: 2932 NW 99th Terrace
Sunrise, FL 33322

FULL NAME OF INVENTOR: Bruce D. Lucas

INVENTOR'S SIGNATURE: _____ DATE: _____

RESIDENCE: 2408 Mill Pond Road
Yorktown Heights, NY 10598

CITIZENSHIP: U.S.A.

POST OFFICE ADDRESS: 2408 Mill Pond Road
Yorktown Heights, NY 10598

DOCKET NUMBER 6169-159

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

J. Rodman Steele, Jr.	Reg. No. 25,931
Gregory A. Nelson	Reg. No. 30,577
Joseph W. Bain	Reg. No. 34,290
Robert J. Sacco	Reg. No. 35,667
Stanley Kim	Reg. No. 42,730
Mark D. Passler	Reg. No. 40,764
Steven Greenberg	Reg. No. 44,725

Send correspondence to Gregory A. Nelson, Quarles & Brady LLP, 222 Lakeview Avenue, Fourth Floor, P.O. Box 3188, West Palm Beach, Florida 33402-3188 and direct all telephone calls to Gregory A. Nelson at (561) 653-5000.

FULL NAME OF INVENTOR: Brett Gavegni

INVENTOR'S SIGNATURE: _____ DATE: _____

RESIDENCE: 2932 NW 99th Terrace
Sunrise, FL 33322

CITIZENSHIP: U.S.A.

POST OFFICE ADDRESS: 2932 NW 99th Terrace
Sunrise, FL 33322

FULL NAME OF INVENTOR: Bruce D. Lucas

INVENTOR'S SIGNATURE: Bruce D. Lucas DATE: 19 JUNE 2000

RESIDENCE: 2408 Mill Pond Road
Yorktown Heights, NY 10598

CITIZENSHIP: U.S.A.

POST OFFICE ADDRESS: 2408 Mill Pond Road
Yorktown Heights, NY 10598

APPENDIX A

Class Hierarchy

class java.lang.Object

- ☐ class.com.ibm.speech.net.www.protocol.https.**HttpsURLStreamHandlerFactory** (implements java.net.URLStreamHandlerFactory)
- ☐ class com.ibm.sslight.SSLContext (implements java.lang.Cloneable)
  - ☐ class com.ibm.speech.net.www.protocol.https.**HttpsClient**
- ☐ class java.net.URLConnection
  - ☐ class java.net.HttpURLConnection
    - ☐ class com.ibm.speech.net.www.protocol.https.**HttpsURLConnection**
- ☐ class java.net.URLStreamHandler
  - ☐ class com.ibm.speech.net.www.protocol.https.**HttpsURLStreamHandler**

APPENDIX B

Class HttpsURLConnection

com.ibm.speech.net.www.protocol.https

Class HttpsURLConnection

```
java.lang.Object
  |
  +--java.net.URLConnection
        |
        +--java.net.HttpURLConnection
              |
              +--com.ibm.speech.net.www.protocol.https.HttpsURLConnection
```

public class **HttpsURLConnection**
extends java.net.HttpURLConnection

The class `HttpsURLConnection` represents a communications link between the application and a URL. Instances of this class can be used both to read from and to write to the resource referenced by the URL. In general, creating a connection to a URL is a multistep process:

`openConnection()`	`connect()`
Manipulate parameters that affect the connection to the remote resource.	Interact with the resource; query header fields and contents.

----------> time

1. The connection object is created by invoking the `openConnection` method on a URL.
2. The setup parameters and general request properties are manipulated.
3. The actual connection to the remote object is made, using the `connect` method.
4. The remote object becomes available. The header fields and the contents of the remote object can be accessed.

The setup parameters are modified using the following methods:

- `setAllowUserInteraction`
- `setDoInput`
- `setDoOutput`
- `setIfModifiedSince`
- `setUseCaches`

and the general request properties are modified using the method:

- `setRequestProperty`

Default values for the `AllowUserInteraction` and `UseCaches` parameters can be set using the methods `setDefaultAllowUserInteraction` and `setDefaultUseCaches`. Default values for general request properties can be set using the `setDefaultRequestProperty` method.

Each of the above `set` methods has a corresponding `get` method to retrieve the value of the parameter or general request property. The specific parameters and general request properties that are applicable are protocol specific.

The following methods are used to access the header fields and the contents after the connection is made to the remote object:

- `getContent`
- `getHeaderField`
- `getInputStream`
- `getOutputStream`

Certain header fields are accessed frequently. The methods:

- `getContentEncoding`
- `getContentLength`
- `getContentType`
- `getDate`
- `getExpiration`
- `getLastModifed`

provide convenient access to these fields. The `getContentType` method is used by the `getContent` method to determine the type of the remote object; subclasses may find it convenient to override the `getContentType` method.

In the common case, all of the pre-connection parameters and general request properties can be ignored: the pre-connection parameters and request properties default to sensible values. For most clients of this interface, there are only two interesting methods: `getInputStream` and `getObject`, which are mirrored in the `URL` class by convenience methods.

`HttpsURLConnection` is a URLConnection with support for HTTPS-specific features. See the spec for details.

Field Summary	
static int	HTTP_ACCEPTED
static int	HTTP_BAD_GATEWAY
static int	HTTP_BAD_METHOD
static int	HTTP_BAD_REQUEST 4XX: client error
static int	HTTP_CLIENT_TIMEOUT
static int	HTTP_CONFLICT
static int	HTTP_CREATED

static int	HTTP_ENTITY_TOO_LARGE
static int	HTTP_FORBIDDEN
static int	HTTP_GATEWAY_TIMEOUT
static int	HTTP_GONE
static int	HTTP_INTERNAL_ERROR
static int	HTTP_LENGTH_REQUIRED
static int	HTTP_MOVED_PERM
static int	HTTP_MOVED_TEMP
static int	HTTP_MULT_CHOICE 3XX: relocation/redirect
static int	HTTP_NO_CONTENT
static int	HTTP_NOT_ACCEPTABLE
static int	HTTP_NOT_AUTHORITATIVE
static int	HTTP_NOT_FOUND
static int	HTTP_NOT_MODIFIED
static int	HTTP_OK 2XX: generally "OK"
static int	HTTP_PARTIAL
static int	HTTP_PAYMENT_REQUIRED
static int	HTTP_PRECON_FAILED
static int	HTTP_PROXY_AUTH
static int	HTTP_REQ_TOO_LONG
static int	HTTP_RESET
static int	HTTP_SEE_OTHER

static int	**HTTP_SERVER_ERROR** 5XX: server error
static int	**HTTP_UNAUTHORIZED**
static int	**HTTP_UNAVAILABLE**
static int	**HTTP_UNSUPPORTED_TYPE**
static int	**HTTP_USE_PROXY**
static int	**HTTP_VERSION**

Fields inherited from class java.net.HttpURLConnection

HTTP_ACCEPTED, HTTP_BAD_GATEWAY, HTTP_BAD_METHOD, HTTP_BAD_REQUEST, HTTP_CLIENT_TIMEOUT, HTTP_CONFLICT, HTTP_CREATED, HTTP_ENTITY_TOO_LARGE, HTTP_FORBIDDEN, HTTP_GATEWAY_TIMEOUT, HTTP_GONE, HTTP_INTERNAL_ERROR, HTTP_LENGTH_REQUIRED, HTTP_MOVED_PERM, HTTP_MOVED_TEMP, HTTP_MULT_CHOICE, HTTP_NO_CONTENT, HTTP_NOT_ACCEPTABLE, HTTP_NOT_AUTHORITATIVE, HTTP_NOT_FOUND, HTTP_NOT_MODIFIED, HTTP_OK, HTTP_PARTIAL, HTTP_PAYMENT_REQUIRED, HTTP_PRECON_FAILED, HTTP_PROXY_AUTH, HTTP_REQ_TOO_LONG, HTTP_RESET, HTTP_SEE_OTHER, HTTP_SERVER_ERROR, HTTP_UNAUTHORIZED, HTTP_UNAVAILABLE, HTTP_UNSUPPORTED_TYPE, HTTP_USE_PROXY, HTTP_VERSION, method, responseCode, responseMessage

Fields inherited from class java.net.URLConnection

allowUserInteraction, connected, doInput, doOutput, ifModifiedSince, url, useCaches

Constructor Summary

HttpsURLConnection(java.net.URL u) Creates a new HttpsURLConnection instance to the object referenced by the URL argument with the default debug flag.
HttpsURLConnection(java.net.URL u, boolean dbg) Creates a new HttpsURLConnection instance to the object referenced by the URL argument with a specified debug flag.

Method Summary

void	**connect**() Opens a communications link to the resource referenced by this URL, if such a connection has not already been established.
void	**disconnect**() Close the connection to the server.
boolean	**getAllowUserInteraction**() Returns the value of the allowUserInteraction field for this object.

java.lang.String	**getContentEncoding**() Returns the value of the content-encoding header field.
int	**getContentLength**() Returns the value of the content-length header field.
java.lang.String	**getContentType**() Returns the value of the content-type header field.
long	**getDate**() Returns the value of the date header field.
static boolean	**getDefaultAllowUserInteraction**() Returns the default value of the allowUserInteraction field.
static java.lang.String	**getDefaultRequestProperty**(java.lang.String key) Returns the value of the default request property.
boolean	**getDefaultUseCaches**() Returns the default value of a URLConnection's useCaches flag.
boolean	**getDoInput**() Returns the value of this URLConnection's doInput flag.
boolean	**getDoOutput**() Returns the value of this URLConnection's doOutput flag.
java.io.InputStream	**getErrorStream**() Returns the error stream if the connection failed but the server sent useful data nonetheless.
long	**getExpiration**() Returns the value of the expires header field.
static java.net.FileNameMap	**getFileNameMap**() Returns the FileNameMap.
static boolean	**getFollowRedirects**()
java.lang.String	**getHeaderField**(int n) Returns the value for the n^{th} header field.
java.lang.String	**getHeaderField**(java.lang.String name) Returns the name of the specified header field.
long	**getHeaderFieldDate**(java.lang.String name, long Default) Returns the value of the named field parsed as date.
int	**getHeaderFieldInt**(java.lang.String name, int Default) Returns the value of the named field parsed as a number.
java.lang.String	**getHeaderFieldKey**(int n) Returns the key for the n^{th} header field.
java.lang.String	**getHTTPHeader**() Returns the entire of the HTTP header received from server request.
long	**getIfModifiedSince**() Returns the value of this object's ifModifiedSince field.
java.io.InputStream	**getInputStream**() Returns an input stream that reads from this open connection.

long	**getLastModified**() Returns the value of the last-modified header field.
java.io.OutputStream	**getOutputStream**() Returns an output stream that writes to this connection.
java.security.Permission	**getPermission**() Returns a permission object representing the permission necessary to make the connection represented by this object.
java.lang.String	**getRequestMethod**() Get the request method.
java.lang.String	**getRequestProperty**(java.lang.String key) Returns the value of the named general request property for this connection.
int	**getResponseCode**() Gets HTTP response status.
java.lang.String	**getResponseMessage**() Gets the HTTP response message, if any, returned along with the response code from a server.
java.net.URL	**getURL**() Returns the value of this URLConnection's URL field.
boolean	**getUseCaches**() Returns the value of this URLConnection's useCaches field.
protected static java.lang.String	**guessContentTypeFromName**(java.lang.String fname) Tries to determine the content type of an object, based on the specified "file" component of a URL.
static java.lang.String	**guessContentTypeFromStream**(java.io.InputStream is) Tries to determine the type of an input stream based on the characters at the beginning of the input stream.
void	**setAllowUserInteraction**(boolean allowuserinteraction) Set the value of the allowUserInteraction field of this URLConnection.
void	**setAsynConnections**(boolean value) Sets the value of the asyncConnectionSet field for the HttpsURLConnection object to the specified value.
static void	**setContentHandlerFactory** (java.net.ContentHandlerFactory fac) Sets the ContentHandlerFactory of an application.
static void	**setDefaultAllowUserInteraction** (boolean defaultallowuserinteraction) Sets the default value of the allowUserInteraction field for all future URLConnection objects to the specified value.
static void	**setDefaultRequestProperty**(java.lang.String key, java.lang.String value) Sets the default value of a general request property.
void	**setDefaultUseCaches**(boolean defaultusecaches) Sets the default value of the useCaches field to the specified value.

void	**setDoInput**(boolean doinput) Sets the value of the doInput field for this URLConnection to the specified value.
void	**setDoOutput**(boolean dooutput) Sets the value of the doOutput field for this URLConnection to the specified value.
void	**setEnableCipherSuites**(java.lang.String cipherSuites) Sets the value of the enabledCipherSuites field for the HttpsURLConnection object to the specified value.
void	**setEnabledCompressionMethods**(java.lang.String methods) Sets the value of the enabledCompressionMethods field for the HttpsURLConnection object to the specified value.
static void	**setFileNameMap**(java.net.FileNameMap map) Sets the FileNameMap.
static void	**setFollowRedirects**(boolean set) Sets whether HTTP redirects (requests with response code 3xx) should be automatically followed by this class.
void	**setIfModifiedSince**(long ifmodifiedsince) Sets the value of the ifModifiedSince field of this URLConnection to the specified value.
void	**setKeyRingDatabase**(java.lang.String name) Sets the value of the keyRingDatabase field for the HttpsURLConnection object to the specified value.
void	**setRequestMethod**(java.lang.String method) Set the method for the URL request, one of: GET POST HEAD OPTIONS PUT DELETE TRACE are legal, subject to protocol restrictions.
void	**setRequestProperty**(java.lang.String key, java.lang.String value) Sets the general request property.
void	**setTimeout**(int seconds) Sets the value of the timeout field for the HttpsURLConnection object to the specified value.
void	**setUseCaches**(boolean usecaches) Sets the value of the useCaches field of this URLConnection to the specified value.
java.lang.String	**toString**() Returns a String representation of this URL connection.
boolean	**usingProxy**() Indicates if the connection is going through a proxy.

Methods inherited from class java.net.URLConnection
getContent

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

HTTP_OK

```
public static final int HTTP_OK
```

2XX: generally "OK"

HTTP_CREATED

```
public static final int HTTP_CREATED
```

HTTP_ACCEPTED

```
public static final int HTTP_ACCEPTED
```

HTTP_NOT_AUTHORITATIVE

```
public static final int HTTP_NOT_AUTHORITATIVE
```

HTTP_NO_CONTENT

```
public static final int HTTP_NO_CONTENT
```

HTTP_RESET

```
public static final int HTTP_RESET
```

HTTP_PARTIAL

```
public static final int HTTP_PARTIAL
```

HTTP_MULT_CHOICE

```
public static final int HTTP_MULT_CHOICE
```

3XX: relocation/redirect

HTTP_MOVED_PERM

public static final int **HTTP_MOVED_PERM**

HTTP_MOVED_TEMP

public static final int **HTTP_MOVED_TEMP**

HTTP_SEE_OTHER

public static final int **HTTP_SEE_OTHER**

HTTP_NOT_MODIFIED

public static final int **HTTP_NOT_MODIFIED**

HTTP_USE_PROXY

public static final int **HTTP_USE_PROXY**

HTTP_BAD_REQUEST

public static final int **HTTP_BAD_REQUEST**

4XX: client error

HTTP_UNAUTHORIZED

public static final int **HTTP_UNAUTHORIZED**

HTTP_PAYMENT_REQUIRED

public static final int **HTTP_PAYMENT_REQUIRED**

HTTP_FORBIDDEN

public static final int **HTTP_FORBIDDEN**

HTTP_NOT_FOUND

```
public static final int HTTP_NOT_FOUND
```

HTTP_BAD_METHOD

```
public static final int HTTP_BAD_METHOD
```

HTTP_NOT_ACCEPTABLE

```
public static final int HTTP_NOT_ACCEPTABLE
```

HTTP_PROXY_AUTH

```
public static final int HTTP_PROXY_AUTH
```

HTTP_CLIENT_TIMEOUT

```
public static final int HTTP_CLIENT_TIMEOUT
```

HTTP_CONFLICT

```
public static final int HTTP_CONFLICT
```

HTTP_GONE

```
public static final int HTTP_GONE
```

HTTP_LENGTH_REQUIRED

```
public static final int HTTP_LENGTH_REQUIRED
```

HTTP_PRECON_FAILED

```
public static final int HTTP_PRECON_FAILED
```

HTTP_ENTITY_TOO_LARGE

```
public static final int HTTP_ENTITY_TOO_LARGE
```

HTTP_REQ_TOO_LONG

```
public static final int HTTP_REQ_TOO_LONG
```

HTTP_UNSUPPORTED_TYPE

```
public static final int HTTP_UNSUPPORTED_TYPE
```

HTTP_SERVER_ERROR

```
public static final int HTTP_SERVER_ERROR
```

5XX: server error

HTTP_INTERNAL_ERROR

```
public static final int HTTP_INTERNAL_ERROR
```

HTTP_BAD_GATEWAY

```
public static final int HTTP_BAD_GATEWAY
```

HTTP_UNAVAILABLE

```
public static final int HTTP_UNAVAILABLE
```

HTTP_GATEWAY_TIMEOUT

```
public static final int HTTP_GATEWAY_TIMEOUT
```

HTTP_VERSION

```
public static final int HTTP_VERSION
```

Constructor Detail

HttpsURLConnection

```
public HttpsURLConnection(java.net.URL u)
```

Creates a new `HttpsURLConnection` instance to the object referenced by the `URL` argument with the default debug flag.
Parameters:
`u` - the URL that this connects to.

HttpsURLConnection

```
public HttpsURLConnection(java.net.URL u,
                          boolean dbg)
```

Creates a new `HttpsURLConnection` instance to the object referenced by the `URL` argument with a specified debug flag.
Parameters:
`u` - the URL that this connects to.
`dbg` - dbg the boolean value of turning the debug option ON/OFF.

Method Detail

getInputStream

```
public java.io.InputStream getInputStream()
                                  throws java.io.IOException
```

Returns an input stream that reads from this open connection.
Overrides:
getInputStream in class java.net.URLConnection
Returns:
an input stream that reads from this open connection.
Throws:
java.io.IOException - if an I/O error occurs while creating the input stream.
UnknownServiceException - if the protocol does not support input.

---

getOutputStream

```
public java.io.OutputStream getOutputStream()
                                   throws java.io.IOException
```

Returns an output stream that writes to this connection.
Overrides:
getOutputStream in class java.net.URLConnection
Returns:
an output stream that writes to this connection.
Throws:
java.io.IOException - if an I/O error occurs while creating the output stream.
UnknownServiceException - if the protocol does not support output.

---

disconnect

```
public void disconnect()
```

Close the connection to the server.
Overrides:
disconnect in class java.net.HttpURLConnection

---

usingProxy

```
public boolean usingProxy()
```

Indicates if the connection is going through a proxy.
Overrides:
usingProxy in class java.net.HttpURLConnection

---

connect

```
public void connect()
            throws java.io.IOException
```

Opens a communications link to the resource referenced by this URL, if such a connection has not already been established.

If the `connect` method is called when the connection has already been opened (indicated by the `connected` field having the value `true`), the call is ignored.

HttpsURLConnection objects go through two phases: first they are created, then they are connected. After being created, and before being connected, various options can be specified (e.g., doInput and UseCaches). After connecting, it is an error to try to set them. Operations that depend on being connected, like getContentLength, will implicitly perform the connection, if necessary.

Overrides:
connect in class java.net.URLConnection
Throws:
java.io.IOException - if an I/O error occurs while opening the connection.

---

setRequestMethod

```
public void setRequestMethod(java.lang.String method)
                    throws java.net.ProtocolException
```

Set the method for the URL request, one of:

- GET
- POST
- HEAD

- OPTIONS
- PUT
- DELETE
- TRACE

are legal, subject to protocol restrictions. The default method is GET.

Overrides:

setRequestMethod in class java.net.HttpURLConnection

Throws:

java.net.ProtocolException - if the method cannot be reset or if the requested method isn't valid for HTTP.

---

setRequestProperty

```
public void setRequestProperty(java.lang.String key,
                               java.lang.String value)
```

Sets the general request property.

Overrides:

setRequestProperty in class java.net.URLConnection

Parameters:

`key` - the keyword by which the request is known (e.g., "`accept`").
`value` - the value associated with it.

---

getHeaderField

```
public java.lang.String getHeaderField(java.lang.String name)
```

Returns the name of the specified header field.

Overrides:

getHeaderField in class java.net.URLConnection

Parameters:

`name` - the name of a header field.

Returns:

the value of the named header field, or `null` if there is no such field in the header.

---

getHTTPHeader

```
public java.lang.String getHTTPHeader()
```

Returns the entire of the HTTP header received from server request.

Returns:

the value of the `httpHeader` field for this object.

---

getAllowUserInteraction

```
public boolean getAllowUserInteraction()
```

Returns the value of the `allowUserInteraction` field for this object.

Overrides:
getAllowUserInteraction in class java.net.URLConnection
Returns:
the value of the `allowUserInteraction` field for this object.

---

getContentEncoding

```
public java.lang.String getContentEncoding()
```

Returns the value of the `content-encoding` header field.
Overrides:
getContentEncoding in class java.net.URLConnection
Returns:
the content encoding of the resource that the URL references, or `null` if not known.

---

getContentLength

```
public int getContentLength()
```

Returns the value of the `content-length` header field.
Overrides:
getContentLength in class java.net.URLConnection
Returns:
the content length of the resource that this connection's URL references, or `-1` if the content length is not known.

---

getContentType

```
public java.lang.String getContentType()
```

Returns the value of the `content-type` header field.
Overrides:
getContentType in class java.net.URLConnection
Returns:
the content type of the resource that the URL references, or `null` if not known.

---

getDate

```
public long getDate()
```

Returns the value of the `date` header field.
Overrides:
getDate in class java.net.URLConnection
Returns:
the sending date of the resource that the URL references, or `0` if not known. The value returned is the number of milliseconds since January 1, 1970 GMT.

---

getDefaultAllowUserInteraction

```
public static boolean getDefaultAllowUserInteraction()
```

Returns the default value of the `allowUserInteraction` field.

Ths default is "sticky", being a part of the static state of all URLConnections. This flag applies to the next, and all following URLConnections that are created.

Returns:
the default value of the `allowUserInteraction` field.

---

getDefaultRequestProperty

```
public static java.lang.String getDefaultRequestProperty(java.lang.String key)
```

Returns the value of the default request property. Default request properties are set for every connection.

Returns:
the value of the default request property for the specified key.

---

getDefaultUseCaches

```
public boolean getDefaultUseCaches()
```

Returns the default value of a `URLConnection`'s `useCaches` flag.

Ths default is "sticky", being a part of the static state of all URLConnections. This flag applies to the next, and all following URLConnections that are created.

Overrides:
getDefaultUseCaches in class java.net.URLConnection
Returns:
the default value of a `URLConnection`'s `useCaches` flag.

---

getDoInput

```
public boolean getDoInput()
```

Returns the value of this `URLConnection`'s `doInput` flag.

Overrides:
getDoInput in class java.net.URLConnection
Returns:
the value of this `URLConnection`'s `doInput` flag.

---

getDoOutput

```
public boolean getDoOutput()
```

Returns the value of this `URLConnection`'s `doOutput` flag.
Overrides:
getDoOutput in class java.net.URLConnection
Returns:
the value of this `URLConnection`'s `doOutput` flag.

---

getExpiration

```
public long getExpiration()
```

Returns the value of the `expires` header field.
Overrides:
getExpiration in class java.net.URLConnection
Returns:
the expiration date of the resource that this URL references, or 0 if not known. The value is the number of milliseconds since January 1, 1970 GMT.

---

getFileNameMap

```
public static java.net.FileNameMap getFileNameMap()
```

Returns the FileNameMap.
Since:
JDK1.2

---

getHeaderField

```
public java.lang.String getHeaderField(int n)
```

Returns the value for the n^{th} header field. It returns `null` if there are fewer than `n` fields.

This method can be used in conjunction with the `getHeaderFieldKey` method to iterate through all the headers in the message.

Overrides:
getHeaderField in class java.net.URLConnection
Parameters:
`n` - an index.
Returns:
the value of the n^{th} header field.

---

getHeaderFieldDate

```
public long getHeaderFieldDate(java.lang.String name,
```

```
long Default)
```

Returns the value of the named field parsed as date. The result is the number of milliseconds since January 1, 1970 GMT represented by the named field.

This form of `getHeaderField` exists because some connection types (e.g., `http-ng`) have pre-parsed headers. Classes for that connection type can override this method and short-circuit the parsing.

Overrides:
getHeaderFieldDate in class java.net.URLConnection
Parameters:
`name` - the name of the header field.
`Default` - a default value.
Returns:
the value of the field, parsed as a date. The value of the `Default` argument is returned if the field is missing or malformed.

---

getHeaderFieldInt

```
public int getHeaderFieldInt(java.lang.String name,
                             int Default)
```

Returns the value of the named field parsed as a number.

This form of `getHeaderField` exists because some connection types (e.g., `http-ng`) have pre-parsed headers. Classes for that connection type can override this method and short-circuit the parsing.

Overrides:
getHeaderFieldInt in class java.net.URLConnection
Parameters:
`name` - the name of the header field.
`Default` - the default value.
Returns:
the value of the named field, parsed as an integer. The `Default` value is returned if the field is missing or malformed.

---

getHeaderFieldKey

```
public java.lang.String getHeaderFieldKey(int n)
```

Returns the key for the n^{th} header field.
Overrides:
getHeaderFieldKey in class java.net.URLConnection
Parameters:
`n` - an index.
Returns:
the key for the n^{th} header field, or `null` if there are fewer than `n` fields.

---

getIfModifiedSince

```
public long getIfModifiedSince()
```

Returns the value of this object's `ifModifiedSince` field.
Overrides:
getIfModifiedSince in class java.net.URLConnection
Returns:
the value of this object's `ifModifiedSince` field.

---

getLastModified

```
public long getLastModified()
```

Returns the value of the `last-modified` header field. The result is the number of milliseconds since January 1, 1970 GMT.
Overrides:
getLastModified in class java.net.URLConnection
Returns:
the date the resource referenced by this `URLConnection` was last modified, or 0 if not known.

---

getPermission

```
public java.security.Permission getPermission()
                                    throws java.io.IOException
```

Returns a permission object representing the permission necessary to make the connection represented by this object. This method returns null if no permission is required to make the connection. By default, this method returns `java.security.AllPermission`. Subclasses should override this method and return the permission that best represents the permission required to make a a connection to the URL. For example, a `URLConnection` representing a `file:` URL would return a `java.io.FilePermission` object.

The permission returned may dependent upon the state of the connection. For example, the permission before connecting may be different from that after connecting. For example, an HTTP sever, say foo.com, may redirect the connection to a different host, say bar.com. Before connecting the permission returned by the connection will represent the permission needed to connect to foo.com, while the permission returned after connecting will be to bar.com.

Permissions are generally used for two purposes: to protect caches of objects obtained through URLConnections, and to check the right of a recipient to learn about a particular URL. In the first case, the permission should be obtained *after* the object has been obtained. For example, in an HTTP connection, this will represent the permission to connect to the host from which the data was ultimately fetched. In the second case, the permission should be obtained and tested *before* connecting.

Overrides:
getPermission in class java.net.HttpURLConnection
Returns:

the permission object representing the permission necessary to make the connection represented by this URLConnection.

Throws:

java.io.IOException - if the computation of the permission requires network or file I/O and an exception occurs while computing it.

---

getRequestProperty

```
public java.lang.String getRequestProperty(java.lang.String key)
```

Returns the value of the named general request property for this connection.

Overrides:

getRequestProperty in class java.net.URLConnection

Returns:

the value of the named general request property for this connection.

---

getURL

```
public java.net.URL getURL()
```

Returns the value of this `URLConnection`'s `URL` field.

Overrides:

getURL in class java.net.URLConnection

Returns:

the value of this `URLConnection`'s `URL` field.

---

getUseCaches

```
public boolean getUseCaches()
```

Returns the value of this `URLConnection`'s `useCaches` field.

Overrides:

getUseCaches in class java.net.URLConnection

Returns:

the value of this `URLConnection`'s `useCaches` field.

---

guessContentTypeFromName

```
protected static java.lang.String guessContentTypeFromName(java.lang.String fname)
```

Tries to determine the content type of an object, based on the specified "file" component of a URL. This is a convenience method that can be used by subclasses that override the `getContentType` method.

Parameters:

`fname` - a filename.

Returns:

a guess as to what the content type of the object is, based upon its file name.

---

guessContentTypeFromStream

```
public static java.lang.String guessContentTypeFromStream(java.io.InputStream is)
                                                    throws java.io.IOException
```

Tries to determine the type of an input stream based on the characters at the beginning of the input stream. This method can be used by subclasses that override the `getContentType` method.

Ideally, this routine would not be needed. But many `http` servers return the incorrect content type; in addition, there are many nonstandard extensions. Direct inspection of the bytes to determine the content type is often more accurate than believing the content type claimed by the `http` server.

Parameters:
`is` - an input stream that supports marks.
Returns:
a guess at the content type, or `null` if none can be determined.
Throws:
java.io.IOException - if an I/O error occurs while reading the input stream.

---

setAllowUserInteraction

```
public void setAllowUserInteraction(boolean allowuserinteraction)
```

Set the value of the `allowUserInteraction` field of this `URLConnection`.
Overrides:
setAllowUserInteraction in class java.net.URLConnection
Parameters:
`allowuserinteraction` - the new value.

---

setContentHandlerFactory

```
public static void setContentHandlerFactory(java.net.ContentHandlerFactory fac)
```

Sets the `ContentHandlerFactory` of an application. It can be called at most once by an application.

The `ContentHandlerFactory` instance is used to construct a content handler from a content type

If there is a security manager, this method first calls the security manager's `checkSetFactory` method to ensure the operation is allowed. This could result in a SecurityException.

Parameters:
`fac` - the desired factory.
Throws:
java.lang.Error - if the factory has already been defined.
java.lang.SecurityException - if a security manager exists and its `checkSetFactory` method doesn't allow the operation.

---

setDefaultAllowUserInteraction

```
public static void setDefaultAllowUserInteraction(boolean defaultallowuserinteracti
```

Sets the default value of the `allowUserInteraction` field for all future `URLConnection` objects to the specified value.

Parameters:

`defaultallowuserinteraction` - the new value.

---

setDefaultRequestProperty

```
public static void setDefaultRequestProperty(java.lang.String key,
                                             java.lang.String value)
```

Sets the default value of a general request property. When a `URLConnection` is created, it is initialized with these properties.

Parameters:

`key` - the keyword by which the request is known (e.g., "`accept`").

`value` - the value associated with the key.

---

setDefaultUseCaches

```
public void setDefaultUseCaches(boolean defaultusecaches)
```

Sets the default value of the `useCaches` field to the specified value.

Overrides:

setDefaultUseCaches in class java.net.URLConnection

Parameters:

`defaultusecaches` - the new value.

---

setDoInput

```
public void setDoInput(boolean doinput)
```

Sets the value of the `doInput` field for this `URLConnection` to the specified value.

A URL connection can be used for input and/or output. Set the DoInput flag to true if you intend to use the URL connection for input, false if not. The default is true unless DoOutput is explicitly set to true, in which case DoInput defaults to false.

Overrides:

setDoInput in class java.net.URLConnection

Parameters:

`value` - the new value.

---

setDoOutput

```
public void setDoOutput(boolean dooutput)
```

Sets the value of the `doOutput` field for this `URLConnection` to the specified value.

A URL connection can be used for input and/or output. Set the DoOutput flag to true if you intend to use the URL connection for output, false if not. The default is false.

Overrides:
setDoOutput in class java.net.URLConnection
Parameters:
`value` - the new value.

---

setFileNameMap

```
public static void setFileNameMap(java.net.FileNameMap map)
```

Sets the FileNameMap.

If there is a security manager, this method first calls the security manager's `checkSetFactory` method to ensure the operation is allowed. This could result in a SecurityException.

Parameters:
`map` - the FileNameMap to be set
Throws:
java.lang.SecurityException - if a security manager exists and its `checkSetFactory` method doesn't allow the operation.

---

setIfModifiedSince

```
public void setIfModifiedSince(long ifmodifiedsince)
```

Sets the value of the `ifModifiedSince` field of this `URLConnection` to the specified value.
Overrides:
setIfModifiedSince in class java.net.URLConnection
Parameters:
`value` - the new value.

---

setUseCaches

```
public void setUseCaches(boolean usecaches)
```

Sets the value of the `useCaches` field of this `URLConnection` to the specified value.

Some protocols do caching of documents. Occasionally, it is important to be able to "tunnel through" and ignore the caches (e.g., the "reload" button in a browser). If the UseCaches flag on a connection is true, the connection is allowed to use whatever caches it can. If false, caches

are to be ignored. The default value comes from DefaultUseCaches, which defaults to true.

Overrides:
setUseCaches in class java.net.URLConnection

---

toString

```
public java.lang.String toString()
```

Returns a `String` representation of this URL connection.
Overrides:
toString in class java.net.URLConnection
Returns:
a string representation of this `URLConnection`.

---

getErrorStream

```
public java.io.InputStream getErrorStream()
```

Returns the error stream if the connection failed but the server sent useful data nonetheless. The typical example is when an HTTP server responds with a 404, which will cause a FileNotFoundException to be thrown in connect, but the server sent an HTML help page with suggestions as to what to do.

This method will not cause a connection to be initiated. If there the connection was not connected, or if the server did not have an error while connecting or if the server did have an error but there no error data was sent, this method will return null. This is the default.

Overrides:
getErrorStream in class java.net.HttpURLConnection
Returns:
an error stream if any, null if there have been no errors, the connection is not connected or the server sent no useful data.

---

getFollowRedirects

```
public static boolean getFollowRedirects()
```

---

getRequestMethod

```
public java.lang.String getRequestMethod()
```

Get the request method.
Overrides:
getRequestMethod in class java.net.HttpURLConnection

---

getResponseCode

```
public int getResponseCode()
                    throws java.io.IOException
```

Gets HTTP response status. From responses like:

```
HTTP/1.0 200 OK
HTTP/1.0 401 Unauthorized
```

Extracts the ints 200 and 401 respectively. Returns -1 if none can be discerned from the response (i.e., the response is not valid HTTP).

Overrides:

getResponseCode in class java.net.HttpURLConnection

Throws:

java.io.IOException - if an error occurred connecting to the server.

---

getResponseMessage

```
public java.lang.String getResponseMessage()
                                    throws java.io.IOException
```

Gets the HTTP response message, if any, returned along with the response code from a server. From responses like:

```
HTTP/1.0 200 OK
HTTP/1.0 404 Not Found
```

Extracts the Strings "OK" and "Not Found" respectively. Returns null if none could be discerned from the responses (the result was not valid HTTP).

Overrides:

getResponseMessage in class java.net.HttpURLConnection

Throws:

java.io.IOException - if an error occurred connecting to the server.

---

setFollowRedirects

```
public static void setFollowRedirects(boolean set)
```

Sets whether HTTP redirects (requests with response code 3xx) should be automatically followed by this class. True by default. Applets cannot change this variable.

If there is a security manager, this method first calls the security manager's `checkSetFactory` method to ensure the operation is allowed. This could result in a SecurityException.

Throws:

java.lang.SecurityException - if a security manager exists and its `checkSetFactory` method doesn't allow the operation.

setKeyRingDatabase

```
public void setKeyRingDatabase(java.lang.String name)
```

Sets the value of the `keyRingDatabase` field for the `HttpsURLConnection` object to the specified value. SSL specific API extension.

Parameters:

`name` - the new value.

setTimeout

```
public void setTimeout(int seconds)
```

Sets the value of the `timeout` field for the `HttpsURLConnection` object to the specified value. SSL specific API extension.

Parameters:

`seconds` - the new value.

setAsynConnections

```
public void setAsynConnections(boolean value)
```

Sets the value of the `asyncConnectionSet` field for the `HttpsURLConnection` object to the specified value. SSL specific API extension.

Parameters:

`value` - the new value.

setEnabledCompressionMethods

```
public void setEnabledCompressionMethods(java.lang.String methods)
```

Sets the value of the `enabledCompressionMethods` field for the `HttpsURLConnection` object to the specified value. SSL specific API extension.

Parameters:

`methods` - the new value.

setEnableCipherSuites

```
public void setEnableCipherSuites(java.lang.String cipherSuites)
```

Sets the value of the `enabledCipherSuites` field for the `HttpsURLConnection` object to the specified value. SSL specific API extension.

Parameters:

`cipherSuites` - the new value.

APPENDIX C

Class HttpsClient

com.ibm.speech.net.www.protocol.https

Class HttpsClient

```
java.lang.Object
  |
  +--com.ibm.sslight.SSLContext
        |
        +--com.ibm.speech.net.www.protocol.https.HttpsClient
```

public class **HttpsClient**
extends com.ibm.sslight.SSLContext

Inner classes inherited from class com.ibm.sslight.SSLContext
`com.ibm.sslight.SSLContext.KeyConstraints`

Field Summary

static int	**CONNECTED** Connection established.
static java.lang.String	**defaultKeyRingDatabase** Default key ring database.
static int	**defaultPort** Default port allocated for HTTPS provided by HTTP RFC.
static int	**NOT_CONNECTED** Connection not established.
static java.lang.String	**sslCertIssuerName** Certificate Issuer Name of SSL established connection.
static java.lang.String	**sslCertOrgName** Certificate's Organization Name of SSL established connection.
static java.lang.String	**sslCipherSuite** Cipher suite of SSL established connection.
static java.lang.String	**sslCompressMethod** Compression method of SSL established connection.
static java.net.InetAddress	**sslInetAddress** java.net.InetAddress of SSL established connection.
static int	**sslPort** Port of SSL established connection.

Fields inherited from class com.ibm.sslight.SSLContext
asyncConnections, CA, clientAuthentication, CONNECT, CONNECTION, debug, SESSION, SITE

Constructor Summary

HttpsClient(java.net.URL u) Creates a new HttpsClient instance with the default debug flag.
HttpsClient(java.net.URL u, boolean debug) Creates a new HttpsURLStreamHandler instance with a specified debug flag.

Method Summary

void	**connect**() Method that calls that sets up SSL connection.
void	**connect**(java.lang.String host, int port) Method that established SSL connection.
void	**disconnect**() Close the connection to the server.
java.lang.String	**getHTTPHeader**() Returns the header from the HTTP request.
java.io.InputStream	**getInputStream**() Returns a input stream that reads from this open connection.
java.io.OutputStream	**getOutputStream**() Returns a java.net.OutputStream that writes to this connection.
int	**getState**() Returns the current state of the connection.
protected boolean	**handleCertificateChain**(com.ibm.sslight.SSLCert[] chain) This method is called by SSL if a certificate chain has to be validated by the SSL protocol but that cannot be done based on the information stored in the public key ring associated with the context or there is no public key ring defined at all.
void	**setAsynConnections**(boolean value) Sets the value of the asyncConnectionSet field for the HttpsClient object to the specified value.
void	**setEnableCipherSuites**(java.lang.String cipherSuites) Sets the value of the enabledCipherSuites field for the HttpsURLConnection object to the specified value.
void	**setEnabledCompressionMethods**(java.lang.String methods) Sets the value of the enabledCompressionMethods field for the HttpsClient object to the specified value.
void	**setKeyRingDatabase**(java.lang.String name) Sets the value of the keyRingDatabase field for the HttpsClient object to the specified value.
void	**setRequestMethod**(java.lang.String method) Sets the value of the method field for the HttpsClient object to the specified value.

void	**setRequestProperties**(java.lang.String properties) Sets the value of the requestProperties field for the HttpsClient object to the specified value.
void	**setTimeout**(int seconds) Sets the value of the timeout field for the HttpsURLConnection object to the specified value.

Methods inherited from class com.ibm.sslight.SSLContext

allowStepUpCryptography, clone, confirmCertificateChain, confirmKeySelection, exportKeyRings, getEnabledCipherSuites, getEnabledCompressionMethods, getKeyRing, getSSLCertByLabel, getTimeout, handleCertificateChain, handleConnection, handleNoPeerCertificate, handleNoSiteCertificate, importCACertificates, importKeyRings, importKeyRings, importSiteCertificates, queryAcceptableKeys, restrictStepUpCryptography, setEnabledCipherSuites, setKeyRing, setTimeout

Methods inherited from class java.lang.Object

equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

sslPort

public static int **sslPort**

Port of SSL established connection.

---

sslInetAddress

public static java.net.InetAddress **sslInetAddress**

java.net.InetAddress of SSL established connection.

---

sslCompressMethod

public static java.lang.String **sslCompressMethod**

Compression method of SSL established connection.

---

sslCipherSuite

public static java.lang.String **sslCipherSuite**

Cipher suite of SSL established connection.

---

sslCertOrgName

```
public static java.lang.String sslCertOrgName
```

Certificate's Organization Name of SSL established connection.

---

sslCertIssuerName

```
public static java.lang.String sslCertIssuerName
```

Certificate Issuer Name of SSL established connection.

NOT_CONNECTED

```
public static final int NOT_CONNECTED
```

Connection not established.

CONNECTED

```
public static final int CONNECTED
```

Connection established.

---

defaultPort

```
public static final int defaultPort
```

Default port allocated for HTTPS provided by HTTP RFC.

---

defaultKeyRingDatabase

```
public static final java.lang.String defaultKeyRingDatabase
```

Default key ring database.

Constructor Detail

HttpsClient

```
public HttpsClient(java.net.URL u)
```

Creates a new `HttpsClient` instance with the default debug flag.
Parameters:
`u` - the java.net.URL associated with the connection.

HttpsClient

```
public HttpsClient(java.net.URL u,
                   boolean debug)
```

Creates a new `HttpsURLStreamHandler` instance with a specified debug flag.
Parameters:
`u` - the java.net.URL associated with the connection.
`dbg` - the boolean value of turning the debug option ON/OFF.

Method Detail

handleCertificateChain

```
protected boolean handleCertificateChain(com.ibm.sslight.SSLCert[] chain)
```

This method is called by SSL if a certificate chain has to be validated by the SSL protocol but that cannot be done based on the information stored in the public key ring associated with the context or there is no public key ring defined at all. It can be overridden in a subclass of SSLContext If not redefined, this method returns false, which means the certificate chain is not verified. In that case the connection establishment is aborted.
Parameters:
`chain` - the `SSLCert[]` assoiciated with the connection. x509chain - the chain of X509.v3 certificates, ordered with the sender's certificate first an the root certificate authority last. correlator - used by the application to associate the handshake with some application defiend Object. This is the same correlator that was used on the SSLSocket constructor or the "accept" for the SSLServerSocket, depending on whether the current role is client or server.

---

getInputStream

```
public java.io.InputStream getInputStream()
                                  throws java.io.IOException
```

Returns a input stream that reads from this open connection.
Returns:
an java.io.InputStream that reads from this open connection.
Throws:
java.io.IOException - if an I/O error occurs while creating the input stream.

---

getOutputStream

```
public java.io.OutputStream getOutputStream()
                                   throws java.io.IOException
```

Returns a java.net.OutputStream that writes to this connection.
Returns:
an output stream that writes to this connection.
Throws:
java.io.IOException - if an I/O error occurs while creating the output stream.

---

connect

```
public void connect()
```

Method that calls that sets up SSL connection.

---

connect

```
public void connect(java.lang.String host,
                    int port)
             throws java.io.IOException
```

Method that established SSL connection.
Parameters:
`host` - the host to request a connection from.
`port` - the port the requested server is listening on.
Throws:
IOExeption - if an error occurs while establishing the connection.

---

disconnect

```
public void disconnect()
```

Close the connection to the server.

---

getState

```
public int getState()
```

Returns the current state of the connection.
Returns:
int the state of the connection 1 if connected, 0 otherwise.

---

getHTTPHeader

```
public java.lang.String getHTTPHeader()
```

Returns the header from the HTTP request.
Returns:

String the header stripped from the connection's java.net.InputStream.

---

setRequestMethod

```
public void setRequestMethod(java.lang.String method)
```

Sets the value of the `method` field for the `HttpsClient` object to the specified value.
Parameters:
`name` - the new value.

---

setRequestProperties

```
public void setRequestProperties(java.lang.String properties)
```

Sets the value of the `requestProperties` field for the `HttpsClient` object to the specified value.
Parameters:
`name` - the new value.

---

setKeyRingDatabase

```
public void setKeyRingDatabase(java.lang.String name)
```

Sets the value of the `keyRingDatabase` field for the `HttpsClient` object to the specified value.
Parameters:
`name` - the new value.

---

setTimeout

```
public void setTimeout(int seconds)
```

Sets the value of the `timeout` field for the `HttpsURLConnection` object to the specified value.
Parameters:
`seconds` - the new value.

---

setAsynConnections

```
public void setAsynConnections(boolean value)
```

Sets the value of the `asyncConnectionSet` field for the `HttpsClient` object to the specified value.
Parameters:
`value` - the new value.

---

setEnabledCompressionMethods

```
public void setEnabledCompressionMethods(java.lang.String methods)
```

Sets the value of the `enabledCompressionMethods` field for the `HttpsClient` object to the specified value.

Overrides:

setEnabledCompressionMethods in class com.ibm.sslight.SSLContext

Parameters:

`methods` - the new value.

---

setEnableCipherSuites

```
public void setEnableCipherSuites(java.lang.String cipherSuites)
```

Sets the value of the `enabledCipherSuites` field for the `HttpsURLConnection` object to the specified value.

Parameters:

`cipherSuites` - the new value.

Class Tree Deprecated Index Help

PREV CLASS **NEXT CLASS**
SUMMARY: INNER | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

APPENDIX D

Class HttpsURLStreamHandlerFactory

com.ibm.speech.net.www.protocol.https

Class HttpsURLStreamHandlerFactory

```
java.lang.Object
  |
  +--com.ibm.speech.net.www.protocol.https.HttpsURLStreamHandlerFactory
```

public class **HttpsURLStreamHandlerFactory**
extends java.lang.Object
implements java.net.URLStreamHandlerFactory

This class implements a factory interface for `URL` stream protocol handlers.

It is used by the `URL` class to create a `HttpsURLStreamHandler` for the `https` protocol.

Constructor Summary

`HttpsURLStreamHandlerFactory()`

Method Summary

`java.net.URLStreamHandler`	`createURLStreamHandler(java.lang.String protocol)` Creates a new `HttpsURLStreamHandler` instance with the specified `https` protocol.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

HttpsURLStreamHandlerFactory

```
public HttpsURLStreamHandlerFactory()
```

Method Detail

createURLStreamHandler

`public java.net.URLStreamHandler` **`createURLStreamHandler`**`(java.lang.String protocol)`

Creates a new `HttpsURLStreamHandler` instance with the specified `https` protocol.

Specified by:

createURLStreamHandler in interface java.net.URLStreamHandlerFactory

Parameters:

`protocol` - the protocol `https`.

Returns:

a `HttpsURLStreamHandler` for the specific protocol.

Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

APPENDIX E

Class HttpsURLStreamHandler

com.ibm.speech.net.www.protocol.https

Class HttpsURLStreamHandler

```
java.lang.Object
  |
  +--java.net.URLStreamHandler
        |
        +--com.ibm.speech.net.www.protocol.https.HttpsURLStreamHandler
```

public class **HttpsURLStreamHandler**
extends java.net.URLStreamHandler

The class `HttpsURLStreamHandler` provides a stream protocol handler for the `https` protocol by implementing SSL (Secure Sockets Layer) 3.0.

In most cases, an instance of a `HttpsURLStreamHandler` subclass is not created directly by an application. Rather, the first time a protocol name is encountered when constructing a `URL`, the appropriate stream protocol handler is automatically loaded.

Constructor Summary

`HttpsURLStreamHandler``()` Creates a new `HttpsURLStreamHandler` instance with the default debug flag.
`HttpsURLStreamHandler``(boolean dbg)` Creates a new `HttpsURLStreamHandler` instance with a specified debug flag.

Method Summary

`java.net.URLConnection`	**`openConnection`**`(java.net.URL u)` Opens a connection to the object referenced by the `URL` argument.

Methods inherited from class java.net.URLStreamHandler

`parseURL, setURL, toExternalForm`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

HttpsURLStreamHandler

```
public HttpsURLStreamHandler()
```

Creates a new `HttpsURLStreamHandler` instance with the default debug flag.

HttpsURLStreamHandler

```
public HttpsURLStreamHandler(boolean dbg)
```

Creates a new `HttpsURLStreamHandler` instance with a specified debug flag.
Parameters:
`dbg` - the boolean value of turning the debug option ON/OFF.

Method Detail

openConnection

```
public java.net.URLConnection openConnection(java.net.URL u)
                                      throws java.io.IOException
```

Opens a connection to the object referenced by the `URL` argument.
Overrides:
openConnection in class java.net.URLStreamHandler
Parameters:
`u` - the URL that this connects to.
Returns:
a `HttpsURLConnection` object for the `URL`.
Throws:
java.io.IOException - if an I/O error occurs while opening the connection.
